**5. Nam Pham’s report**

I will have to change some part of my database before I can transfer the data to the new one.

Main table:

-         Works table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current table:*   |  | | --- | | **All\_works** | | wk-itemalpha  wk-itemnumber | | wk-name  wk-description  wk-type  wk-subtype  wk-author  wk-date\_complete  wk-date\_accquire  wk-insurance\_value  wk-claim  wk-potray\_with | | *New group table:*   |  | | --- | | **Works** | | wk-IDAlpha  wk-IDNumeric | | wk-worksName  wk-type  wk-subtype  wk-Creator  wk-completionDate  wk-acquisitionDate  wk-workDexcription  wk-ownership  wk-question  wk-answer | |

For this table, my current table already look identical to it, including the primary key, although there are still some differences.

The table that the group agree on will not contain the **insurance\_value** value of the item because that value will be record as the temporal data for the new table *WorksInsuranceValue*. So, I will need to add all the current item value to that table and set the same date range for all of them. Because there aren’t any change in the insurance value, so all the end\_date value will be null.

*Test*: I will create a query that choose the idnumber and idalpha from the new database and find it whether the data of the new database catch up. For the insurance value which are now in the new table, I will also make it one part of the query even when it can cause multiple row.

-         Locations table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Galleries** | | lc-name | | lc-dimension  lc-min\_capacity  lc-max\_capacity | | *New:*   |  | | --- | | **Locations** | | lc-name  lc-museumName | | lc-telStartDate  lc-telEndDate  lc-dimension  lc-minNumWorks  lc-maxNumWorks  ls-availableDate | |

For the location table, because we will combine the data of all the group members together, we need to distinguish each member location by adding the attribute **museumName** and make it as a part of the PK. So, even when our locations the same names, it does not violet the PK unique rule. Therefore, I just need to add the name of my museum for each row of location information.

Now, even though my old table already have the attributes like: dimension, min and max capacity, we decided to add more attributes for the locations of the traveling exhibitions later. So, the **telStartdate** and **telEndDate** will be used to tell, for a temporary location, when the exhibition will be held and end at that place. So, because I have stored the date information in the *exhibition\_info* table of my old database, I will use it for this 2 attribute. However, for the locations that are in the main museum, they will never held a traveling exhibition and therefore, this 2 attributes will be null for them.

The **availableDate** is used for when the location is not occupied and can be used. It is most likely the end date of the exhibitions. So, for now I will only need to update this attribute with the date after the exhibitions end, and then later I can create a trigger or a rule that can used this attribute to plan the exhibition in the future.

Test: I will create a query that include the location name to check whether or not the data catch up. The tables that will be used from the old database to check is the galleries table and the temporary location table.

-         Exhibitions table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **exhibitions\_info** | | ex-exhibition\_id | | ex-theme  ex-ex\_descripton  ex-number\_of\_items  ex-start\_date  ex-end\_date  ex-location\_name | | *New:*   |  | | --- | | **exhibitions** | | ex-exhibitionName  ex-exhibitionStartDate  ex-museumName | | ex-istraveling  ex-exhibitionDescription  ex-exhibitonEndDate | |

In my old table, I used the **exhibition\_id** as the set of 3 characters to counter the problem that is 2 exhibitions can have the same name. However, our group decide to go with using the **exhibitionName** and the **exhibitionStartDate** as a part of the PK to counter the problem. We also add the **museumName** attribute to distinguish our exhibitions from each other. So, for me, I have to create a new set of PK and like in the Location tables, I will add my museum name for each row of exhibitions information. I will also have to drop the **exhibition\_id** as I won’t use it in the future.

I also have to drop the number\_of\_items and location\_name as they will be store elsewhere in the database. However, I will have another attribute **istraveling** which is a Boolean. My data will have difference from my group as for a travelling exhibition, they will use it as a whole, but I divided it into 5 exhibitions for each temporary location with the different start date. So, I can add **istraveling** to tell whether or not the 5 exhibitions is actually one or not, but the way I represent the traveling exhibition will be different from my groupmates.

Test: I will create a query that use the museum name to check whether or not the data in the new database catch up or not. The table from the old database that is used to compare is exhibitions\_info table.

Temporal data Tables:   The tests in these tables will be most likely use a query to find out that the data has matched up or not because we only need to change the PK set of them.

-         TempExhibitionsLocations table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Temporary\_location** | | tel-name\_location          FK-lc | | tel-sponsor  tel-address  tel-head\_of\_security  tel-insurance | | *New:*   |  | | --- | | **TempExhibitionsLocations** | | tel-lcname                                               FK-lc  tel-lcMuseumName                                 FK-lc  tel-tempexhibitionLocationStartDate | | tel-sponsor  tel-security  tel-insurance  tel-tempExhibitionLocationEndDate  tel-tempExhibitionLocationAddress | |

In my old table database, I forgot to add the situation that the temporary location can be reused in the future and this table can be used as a temporal data storage. Now, our group decide to add the **tempexhibitionLocationStartDate** attribute to know the time the temporary location is used and also make it a part of the PK. Moreover, because this table is connected to the *TempExhibitionsLocations* table it also need to add the **lcMuseumName** attribute to be a part of the PK and the FK to *locations* table as the PK in the *locations* table is changed. So, what I need to do here is to change the part of add the museum attribute and change the set of PK and the FK in the *Temporary\_location* table.

The **startDate** and **endDate** attributes in this table will have the also have the same value as the one in the *locations* table, so I can easily update them.

-          WorksLocation table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **work\_location** | | wl-itemalpha\_ex          FK-wk    wl-itemnumber\_ex          FK-wk wl-current\_location  wl-date\_in | | wl-date\_out  wl-time\_in  wl-time\_out | | *New:*   |  | | --- | | **WorksLocations** | | wl-lcName                                       FK-lc  wl-wkIDAlpha                                   FK-wk  wl-wkIDNumeric                              FK-wk  wl-lcmuseumName                          FK-lc  wk-workLocationStartDateTime | | wk- workLocationEndDateTime | |

For this table, because of the change of PK set in *locations* table, I will need to add the **lcmuseumName** attribute to connect it with the location table.

The other thing I need to do is to combine the 2 attributes **date\_in** and **time\_in**, so that the attribute can show both date and time. After that I can make the result attribute as a part of the PK in this table. The same will be done for **date\_out** and **time\_out**, but we don’t need it as part of the PK. No further change need to be made for this table.

-         ExhibitionsLocations table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Exhibition\_location** | | el-Ex\_id                            FK-ex  el-current\_location           FK-lc | | el-no\_of\_items | | *New:*   |  | | --- | | **ExhibitionsLocations** | | el-lcName                                FK-lc  el-exName                              FK-ex  el-exStartDate                        FK-ex  el-lcexMuseumName               FK-lc, ex | | el-lctelStartDate  el-lctelEndDate  el-exEndDate | |

Because the set of PK in both *Exhibitions* table and *Locations* table are changed, I will need to add the more attributes to make a new PK. All of the data for these new attributes are already available in *locations* and *exhibitions* and therefore, can be updated from those tables. Because **Ex\_id** is not needed in the new database, I will also drop it.

The no\_of\_items attribute is also not needed for this table, so I will also drop it.

The 2 new special attributes like: **lctelStartDate** and **lctelEndDate** are used to store the date of the traveling exhibition. For this dates, I can find them both in the *exhibitions* table or the *temporary location* table, therefore I can update it from there. However, for the temporary locations, the value of this attribute will be null.

-         ExhibitionsWorks table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Exhibition\_works** | | ew-itemalpha\_ex               FK-wk  ew-itemnumber\_ex            FK-wk  ew-ex\_id                            FK-ex | |  | | *New:*   |  | | --- | | **ExhibitionsWorks** | | ew-exName                          FK-ex  ew-wkIDAlpha                     FK-wk  ew-wkIDNumeric                FK-wk  ew-exStartDate                   FK-ex  ew-exMuseumName          FK-ex | | ew-endDate | |

This table has to be changed due to the change of the PK set from the exhibitions table. So, after I added all the new attributes, I can update the information from the exhibitions table.

-         WorksKeepers table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **donors** | | do-itemalpha\_don                    FK-wk  do-itemnumber\_don                FK-wk | | do-donor\_name | | **loan\_institution** | | lo-itemalpha\_loan                   FK-wk  lo-itemnumber\_loan                FK-wk | | lo-name  lo-address  lo-phonenum  lo-email  lo-start\_date  lo-end\_date | | *New:*   |  | | --- | | **WorksKeepers** | | wkk-workKeeperName  wkk-wkIDNumberic                FK-wk  wkk-wkIDAlpha                      FK-wk  wkk-workKeeperStartDate  wkk-status | | wkk-workKeeperAddress  wkk-workKeeperEndDate  wkk-email  wkkphonenumber | |

The *workskeeper* table is the combination of the donors and *loan\_institution* table. Because this 2 tables have the same attributes and can work in the same way, all we need to do is to add another attribute **status** to know whether or not the items are lend or borrowed. All the data can be taken from this 2 old tables after the merge.

When I do these 2 tables, I didn’t think to make them the temporal data storage, so I only make the item identification to be the PK. However, in this project, making it a temporal data storage mean I have to change the PK set like the above, but all the information can be update easily.

-         Doors table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Current:*   |  | | --- | | **Doors** | | dr-door\_from          FK-lc  dr-door\_to                 FK-lc | | dr-door\_name | | *New:*   |  | | --- | | **Doors** | | dr-lcMuseumName          FK-lc  dr-lcName1                      FK-lc  dr-lcname2                       FK-lc | |  | |

The doors table doesn’t have many major change.

I will need to drop the **doorname** as it is considered unnecessary.

Then, because the PK set in the locations table is changed, I will need to add the **museum name** as part of the table PK. This will also be a way to distinguished the doors between the museum. All the data can be taken from *locations* table.

-         Media table

|  |
| --- |
| **Media** |
| md-wklIdAlpha              FK-wk  md-wklIdNumber          FK-wk  md-material |
|  |

For this table, it is identical to my current one and have the same primary key as foreign key. Therefore, I can transfer my data to this table easily.

New tables:

-         WorksState table

|  |
| --- |
| **WorksState** |
| ws-wkIDNumeric                 FK-wk  ws-wkIDAlpha                      FK-wk  ws-workStateStartDate |
| ws-state  ws-workStateEndDate |

This table is used to check the current state of the item which including: in good condition, stolen, damaged, etc and the time range they are in that condition. So, for all of my current item now, I will just make them into ‘in good condition’ state.

-         WorksInsuranceValue table

|  |
| --- |
| **WorksInsuranceValue** |
| wiv-wkIDNumeric                                          FK-wk  wiv-wkIDAlpha                                               FK-wk  wiv-workInsuranceValueStartDate |
| wiv-insuranceValue  wiv- workInsuranceValueEndDate |

This table will keep the record of the table insurance value that will be changed by the time. However, because all my insurance value have not been changed yet, I will only keep the current insurance and only insert new one when it is changed.